Cats And Dogs Challenge

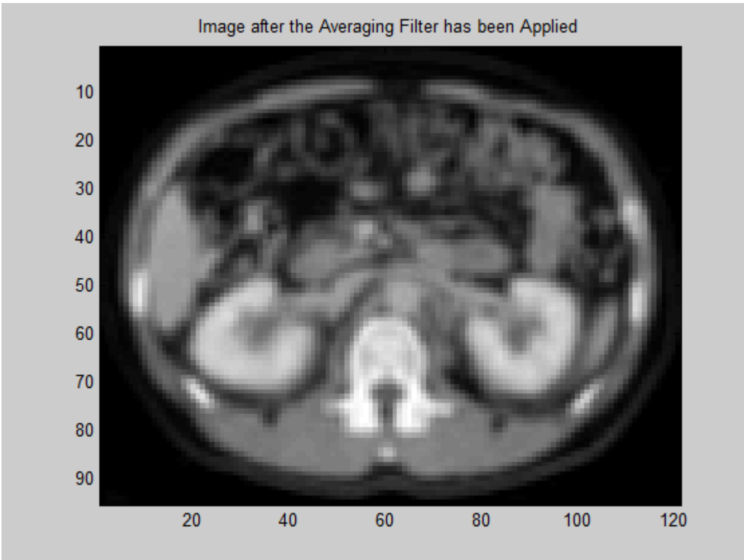
Pattern Recognition and Geometric Data Analysis

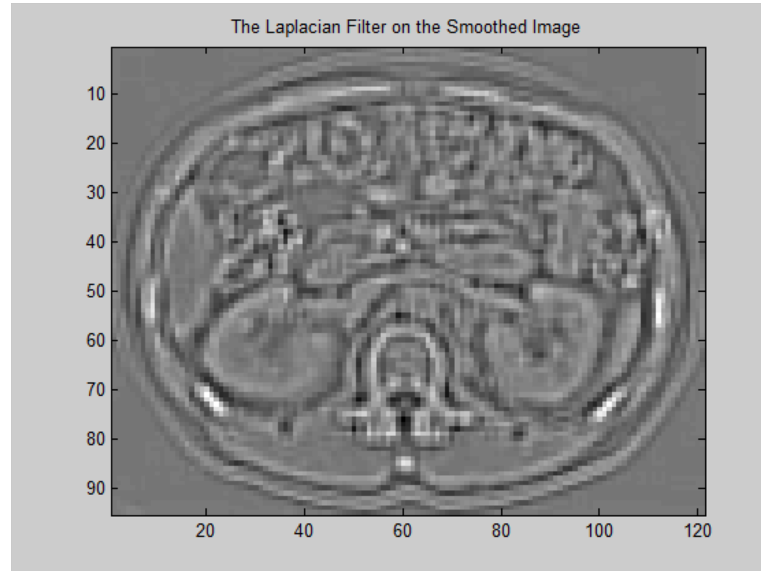
Goal

- Classify Images of Cats and Dogs

- Method:
  - Averaging and Laplacian Filters
  - Principle Component Analysis
  - Fisher Linear Discriminant Analysis

Averaging and Laplacian Filters





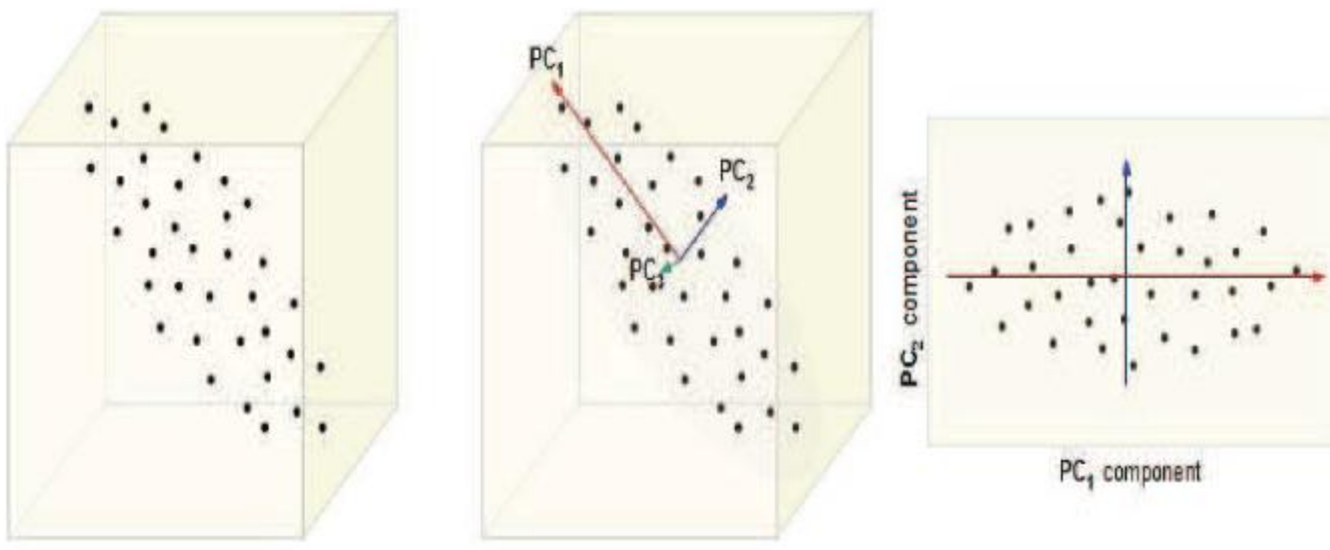
The Averaging Filter is used to make edges in an image smooth

$$AF = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The Laplacian Filter is used for edge detection

$$LF = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

PCA

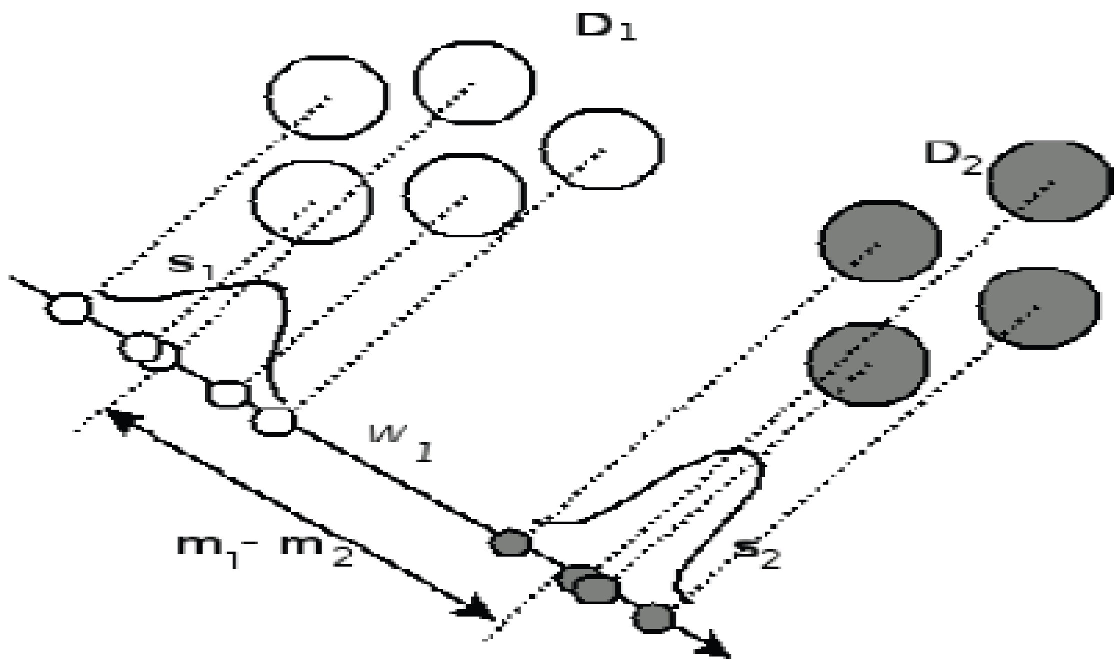


Principle Component Analysis is used for Reduced Dimensionality and Classification

Singular Value Decomposition

$$X = U\Sigma V^T$$

Fischer Linear Discriminant Analysis



LDA (FDA) is used to separate classes

Expansion Coefficients

$$A = \Sigma V^T$$

Results

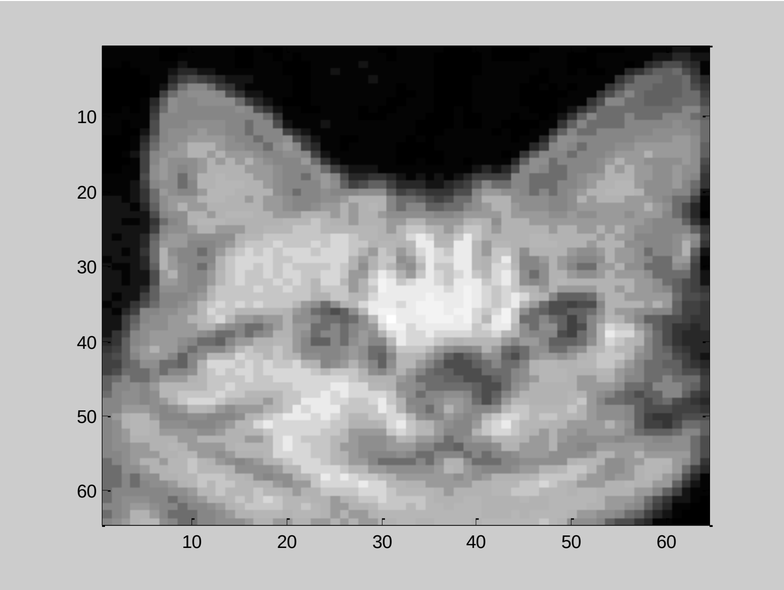


Image of a Cat from the testing set

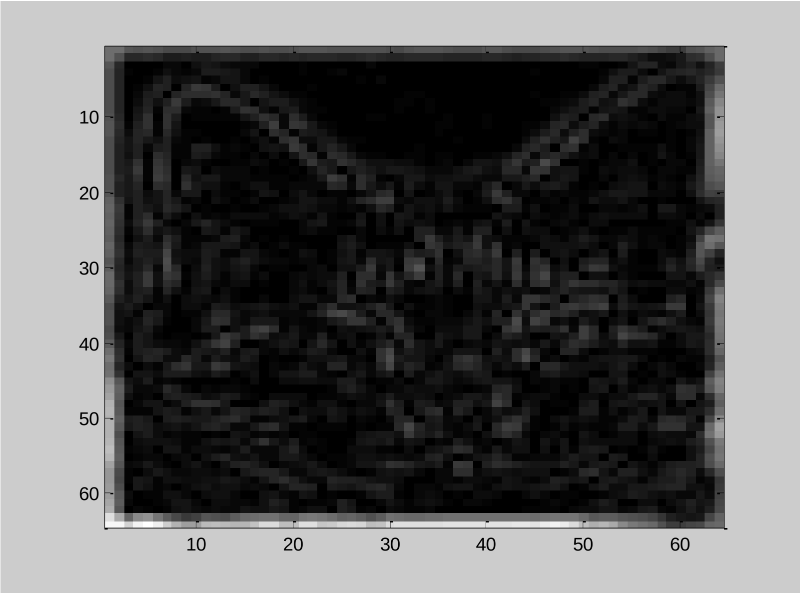
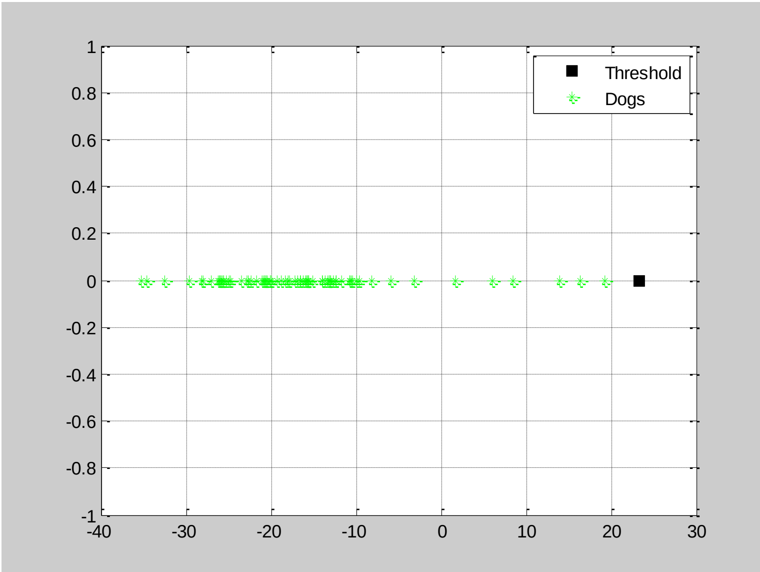
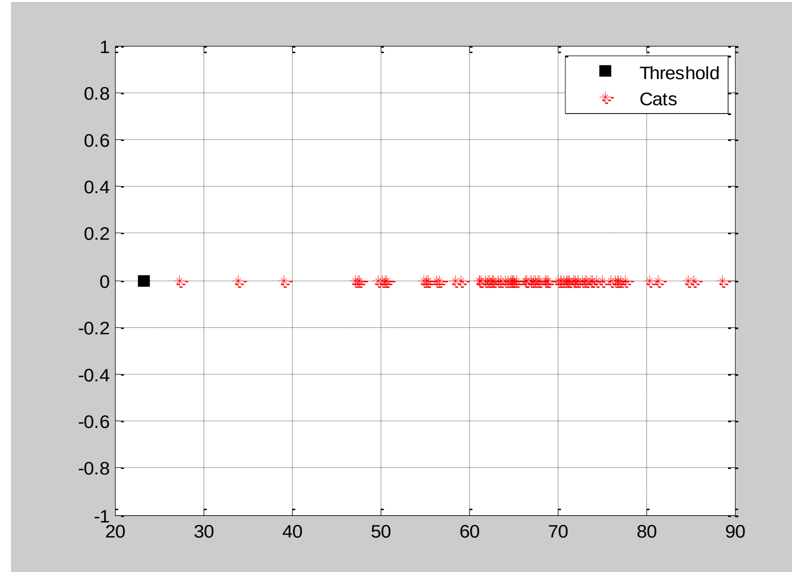


Image of the Cat after the Laplacian Filter has been used and color map has been changed





Result of Fischer Linear Discriminant Analysis. Projection onto an optimal vector, **w**.

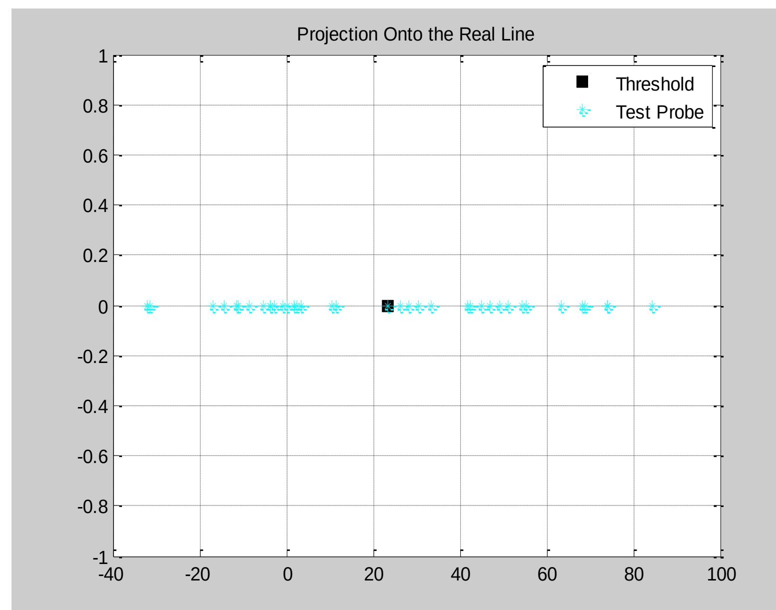


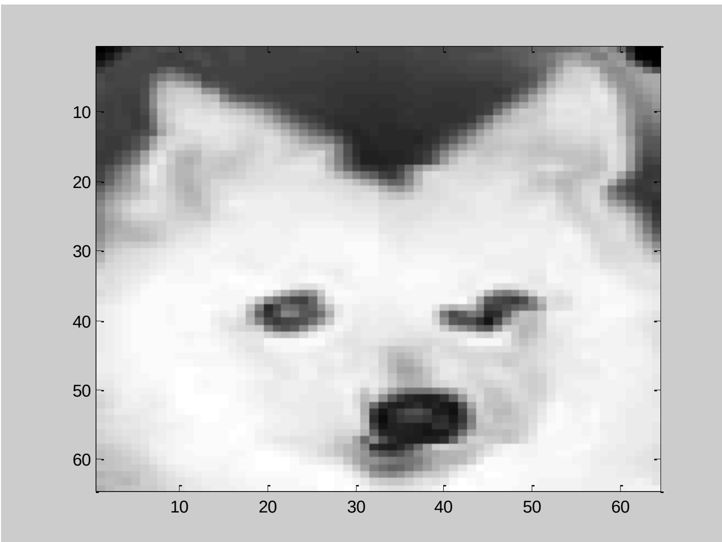
Table of Confusion

	Cats	Dogs
Cats	19	0
Dogs	1	18

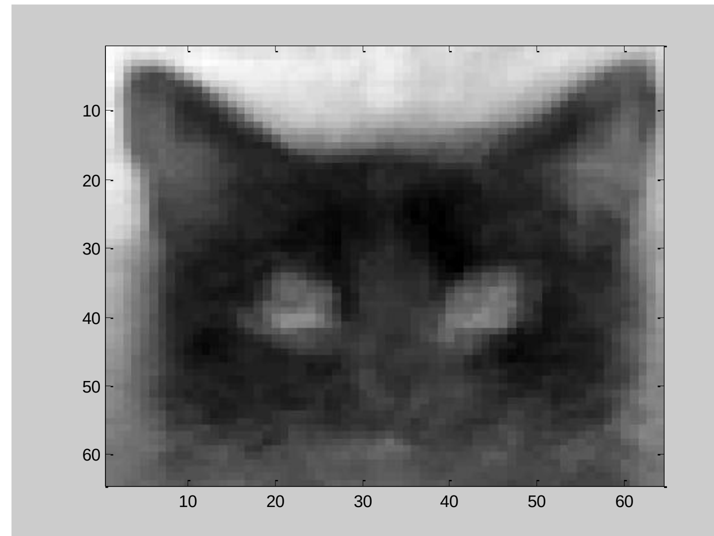
The method gives a 97% classification rate for our data.

Why objects may be misclassified

Misclassified canine



FIDO!!!



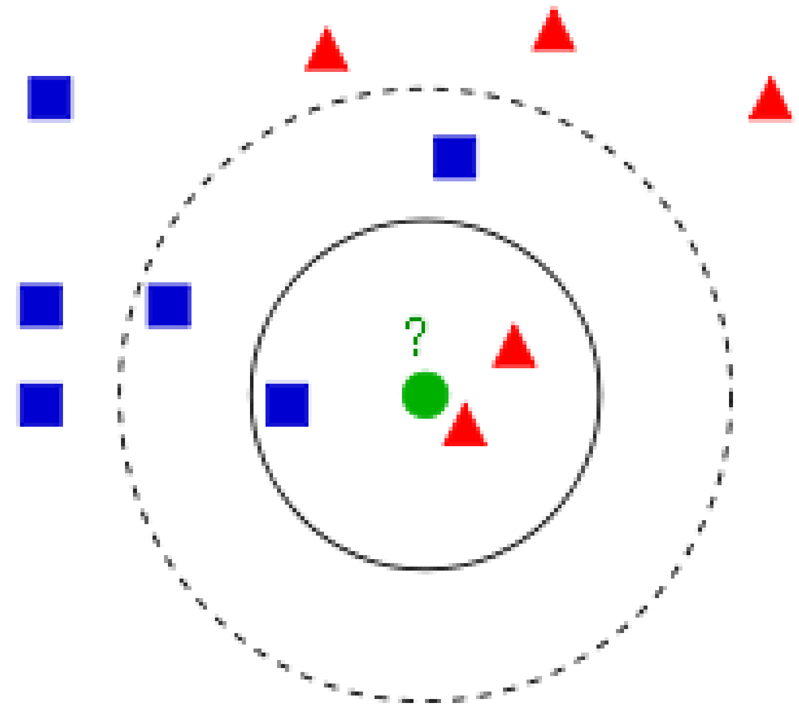
First eigenface of the cats

Classification by nearest neighbors

K-Nearest Neighbor Search

- For a new point y, find $d(x_i, y),\ x_i \in X$
- Take the K smallest values
- Find the 'mode'

- Note: K should be odd



KNN with an Adaptive Metric

- Define a new metric as
  - $d_{new}(y, x_i) = \frac{d(y, x_i)}{r_i}$
  - Where, $r_i = \min_{Y_k \neq Y_i} d(x_k, x_i)$, and Y_k and Y_i represent different classes
- This makes the smallest distance between a training point and another class 1.
- This is not a "true" metric, since $d(x, y) \neq d(y, x)$, but it works
- 2007 – Jigang Wang, Predrag Neskovic, Leon N. Cooper
  - Brown University – Department of Physics, The Institute for Brain and Neural Systems

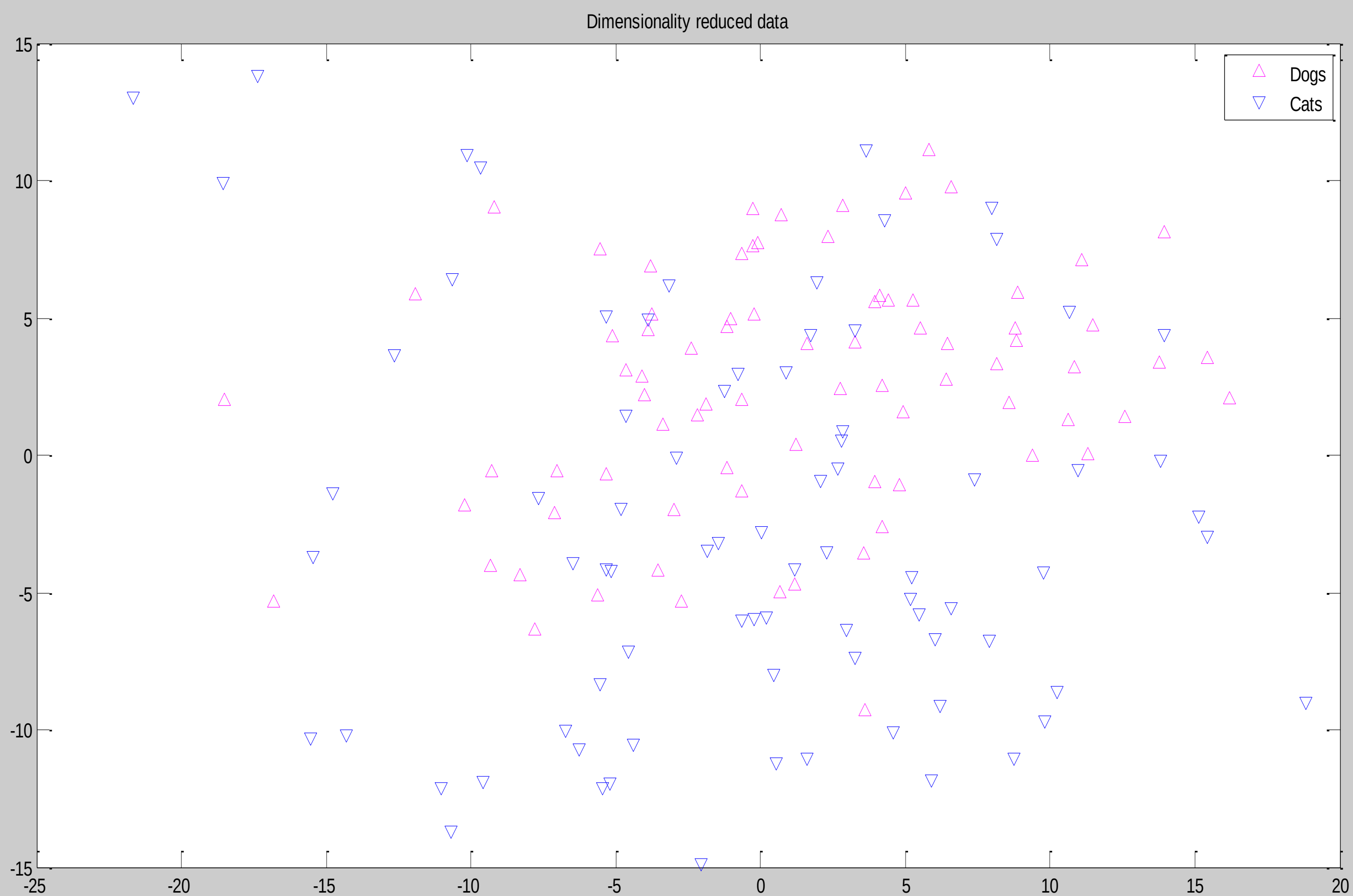
The metrics used with KNN

- Euclidian $d(x, y) = (x - y)(x - y)'$

- Cosine $d(x, y) = 1 - \frac{xy'}{\sqrt{xx'}\sqrt{yy'}}$

- Correlation $d(X, Y) = 1 - \frac{(x-\bar{x})(y-\bar{y})'}{\sqrt{(x-\bar{x})(x-\bar{x})'}\sqrt{(y-\bar{y})(y-\bar{y})'}}$

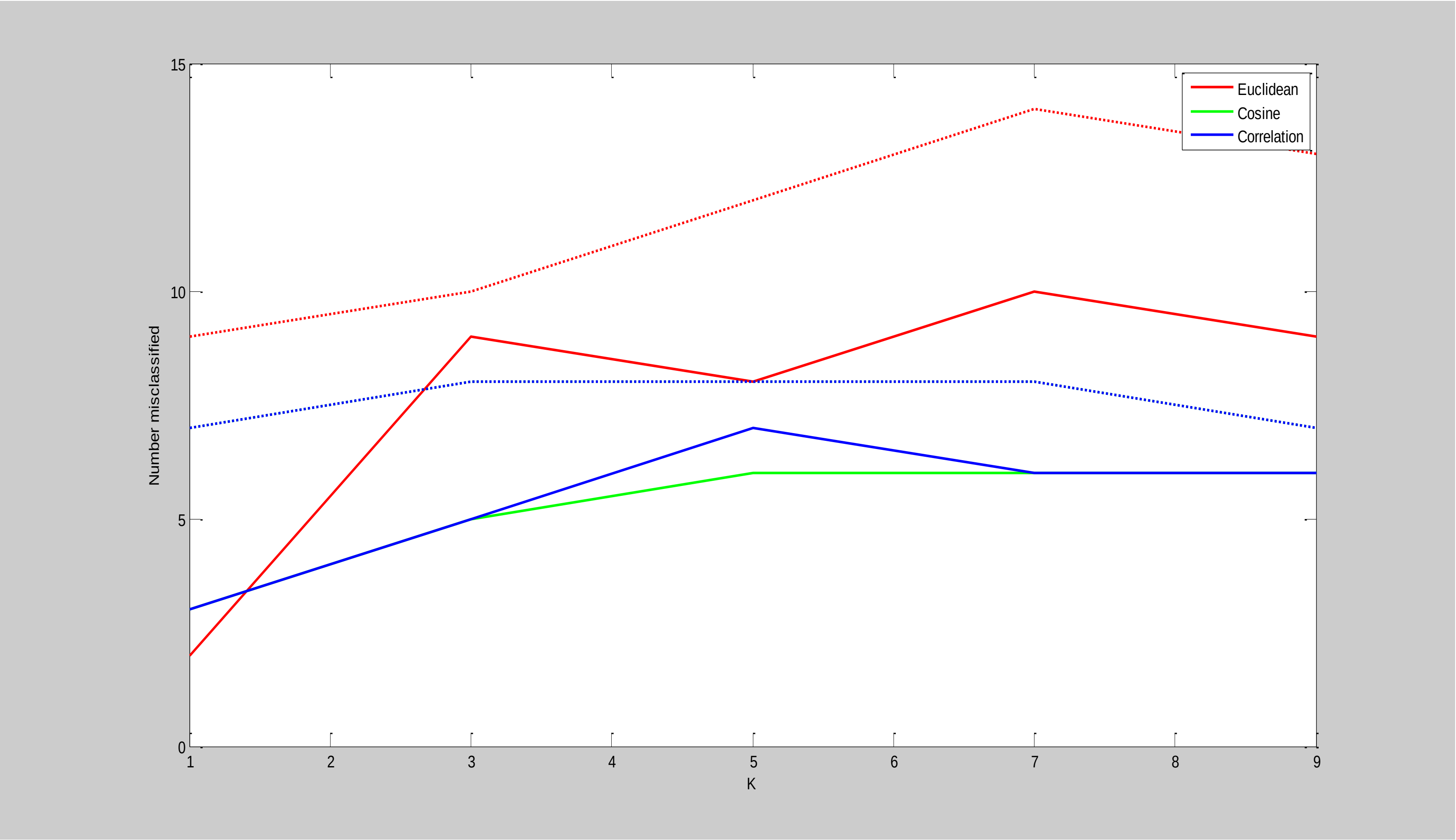
A look at the data



Adaptive KNN Results

- Best results:
  - K=1, Euclidian metric
    - Misclassified 2 dogs as cats
  - K=1, Cosine and Correlation metric
    - Misclassified 3 dogs as cats
- Observations:
  - As K increased, misclassification increased

Adaptive KNN Results



Support Vector Machines

Overview

Linear Learning Machines

Kernel-Induced Feature Spaces

Unsupervised Learning

Master Yoda, I am confused about cats and dogs.

Embarrassing, much to learn we both have.

Supervised Learning

Master Yoda, I am confused about cats and dogs.

Much to learn you have, my young padawan. Explain it once more, I will.

Supervised Learning

In supervised learning, the learning machine is given a *training set* (inputs) with associated known labels (output values). Customarily, input values are in the form of vectors so that the *input space* is a subset of $\mathbb{R}^n$

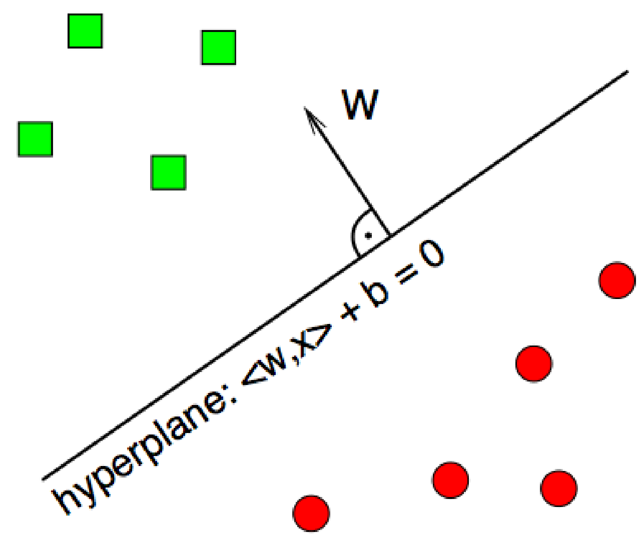
Learning/Training means a decision rule can be found that explains the training set well. (Clearly, this part is easy since labels for the training set are known)

Rosenblatt's Perceptron

First iterative algorithm for learning linear classifications for the perceptron (binary classifier).

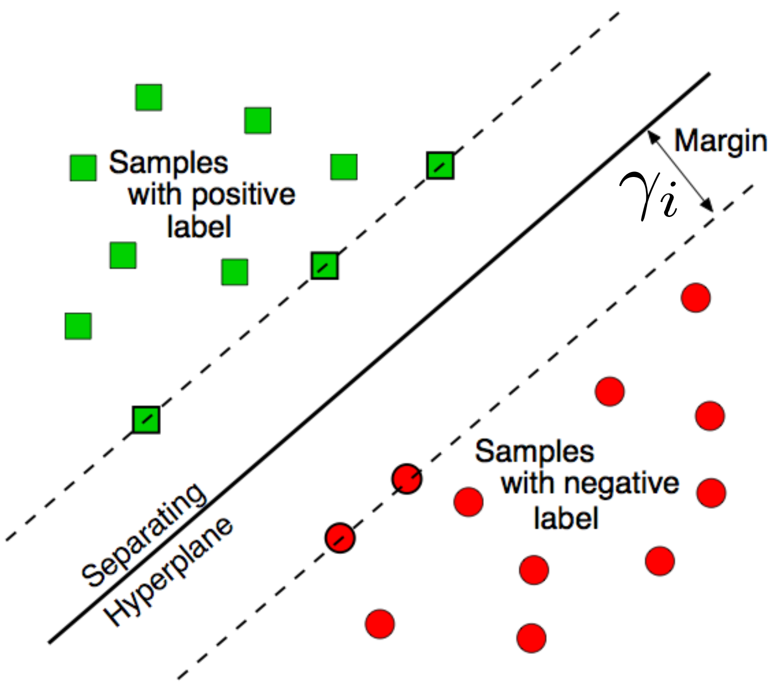
Takes in an initial weight $w_0 = 0$ and adapts at each time a training point is misclassified by current weights. The procedure is guaranteed to converge if there exists a hyperplane that correctly classifies the training data.

Definition. The *functional* margin of an example $(\mathbf{x_i}, y_i)$ with respect to a hyperplane $(\mathbf{w}, b)$ is defined as $\gamma_i = y_i(< \mathbf{w} \cdot \mathbf{x} > +b)$



Linear Classifier (Continued)

If $\gamma_i > 0$, then the classification of $(\mathbf{x_i}, y_i)$ is correct. If the margin is replaced by *geometric margin*, the distribution hyperplane $(\mathbf{w}, b)$ is now a normalized linear function $(\frac{1}{||w||}\mathbf{w}, \frac{1}{||w||}b)$ measuring the Euclidean distances of the points from the decision boundary in the input space. This margin of the training set is now the *maximal margin hyperplane*. The margin is positive for a linearly separable set.



Support Vectors

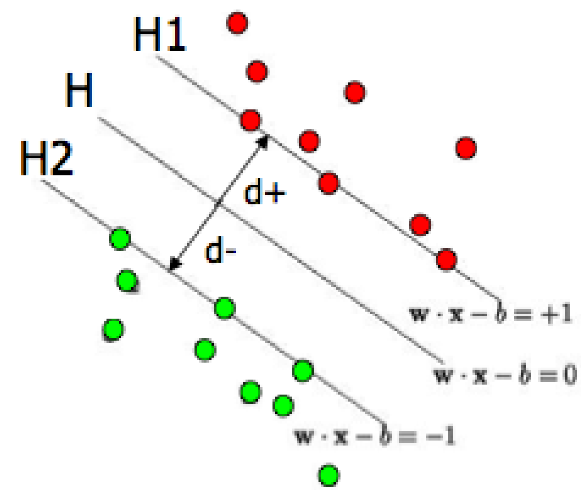
Support Vectors are points nearest to the separating hyperplane.

They determine the position of the hyperplane, while all other points are independent and do not influence the hyperplane.

The weighted sum of these support vectors is the normal vector of the hyperplane.

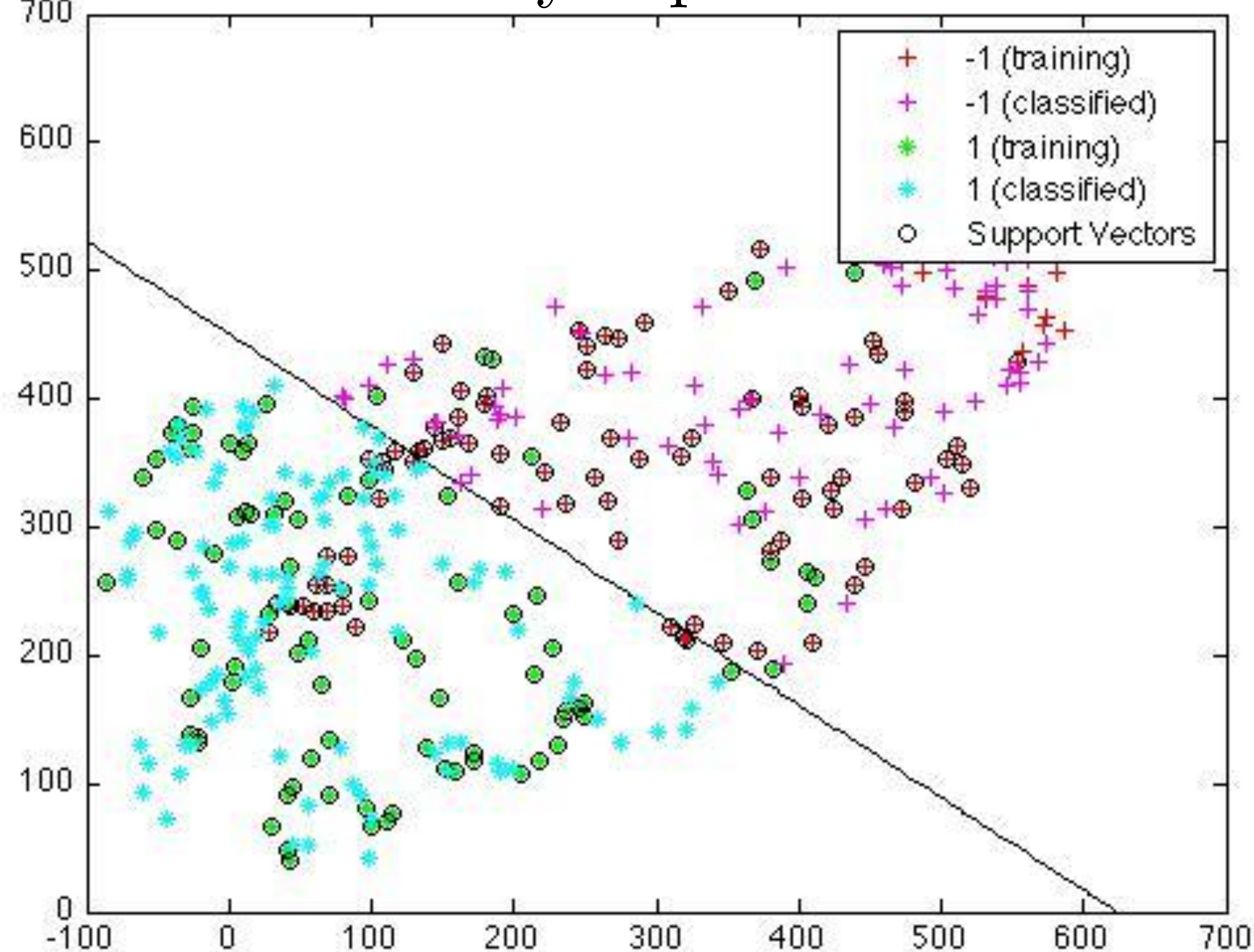
The hyperplane γ, called H in the figure is defined as $< x_i, w > +b \geq +1$ when $y_i = +1$ and $< x_i, w > +b \leq -1$ when $y_i = -1$.
The points that lie on the lines that satisfy the equalities are the support vectors.
From previous slide, the distance between H_1 and H_2 is $2/||w||$. So, in order to maximize margin, seek to minimize $||w||$ with the condition there are no data points between H_1 and H_2. That is

Taking $< x_i, w > +b \geq +1$ when $y_i = +1$
and $< x_i, w > +b \leq -1$ when $y_i = -1$,
combine to give $y_i < x_1, w > \geq 1$.
Thus, as previously stated γ needs to be positive in order for correct classification



Examples of minimizing $||w||$ – Lagrangian, Quadratic Programming

Non-Linearly Separable Data

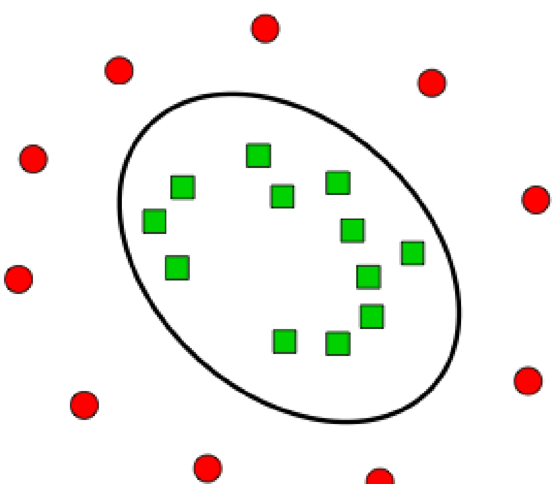


Learning in a Feature Space (higher dimensional space)

Let X be the *input space*. Then, the suitable representation for the data quantities referred to as features is chosen by a mapping $\phi : X \to F$. The space $F = \{\phi(\mathbf{x}) : \mathbf{x} \in X\}$ is called the *feature space*.

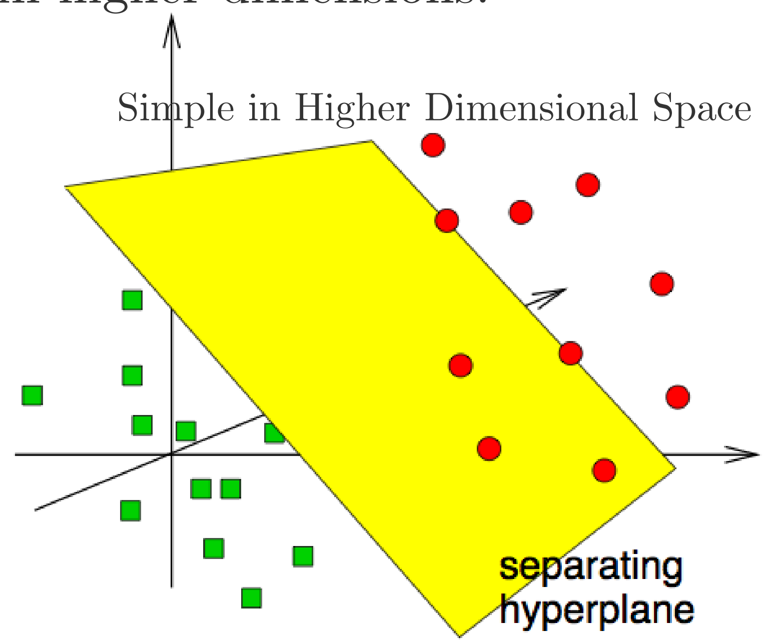
If classification is easier in higher dimensions, we want to build a maximal hyperplane there. Its construction depends on inner products, which will be evaluated in the higher dimensions

Computationally, this can become, costly if the dimensions are high. However, there exists a loophole. We use a kernel function that lives in low dimensions but behaves like an inner product in higher dimensions.



Complex in Low Dimensional Space

$$\phi : X \to F$$



Simple in Higher Dimensional Space

Kernel Functions

The *Kernel* is a function $\mathcal{K}$, such that for all $\mathbf{x}, \mathbf{y} \in X$, $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$, where ϕ maps from the input space X to the (inner product) feature space F.

Given a kernel function, the decision rule is now,

$$f(\mathbf{x}) = \sum_{i=0}^{\mathcal{L}} \alpha_i y_i \mathcal{K}(\mathbf{x_i}, \mathbf{x}) + b$$

for $\mathcal{L}$ iterations of the Kernel

Thus, the maximal margin hyperplane is generated by the Kernel Function in the input space.

Kernel Examples

linear	$\mathcal{K}(x,y) = < x, y >$
polynomial	$\mathcal{K}(x,y) = (\gamma < x, y > + c_0)^d$
radial basis function	$\mathcal{K}(x,y) = \exp(-\gamma \|\|x - y\|\|^2)$

MATLAB Implementation Parameters

Principle Component Analysis

SVMTRAIN

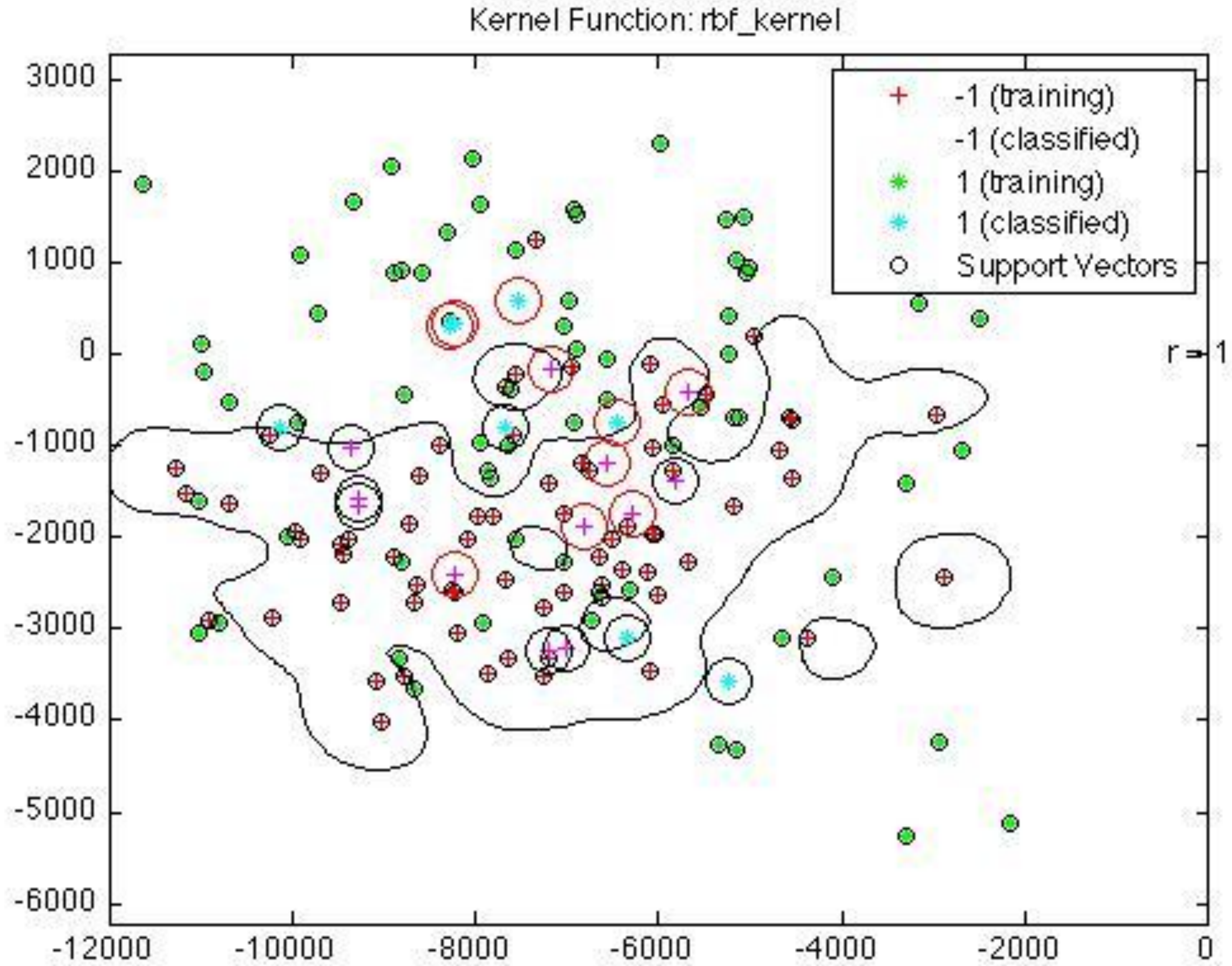
Kernel rbf:

$\gamma = 1$ and $\gamma = 0.2$

box constraint C for for vector α

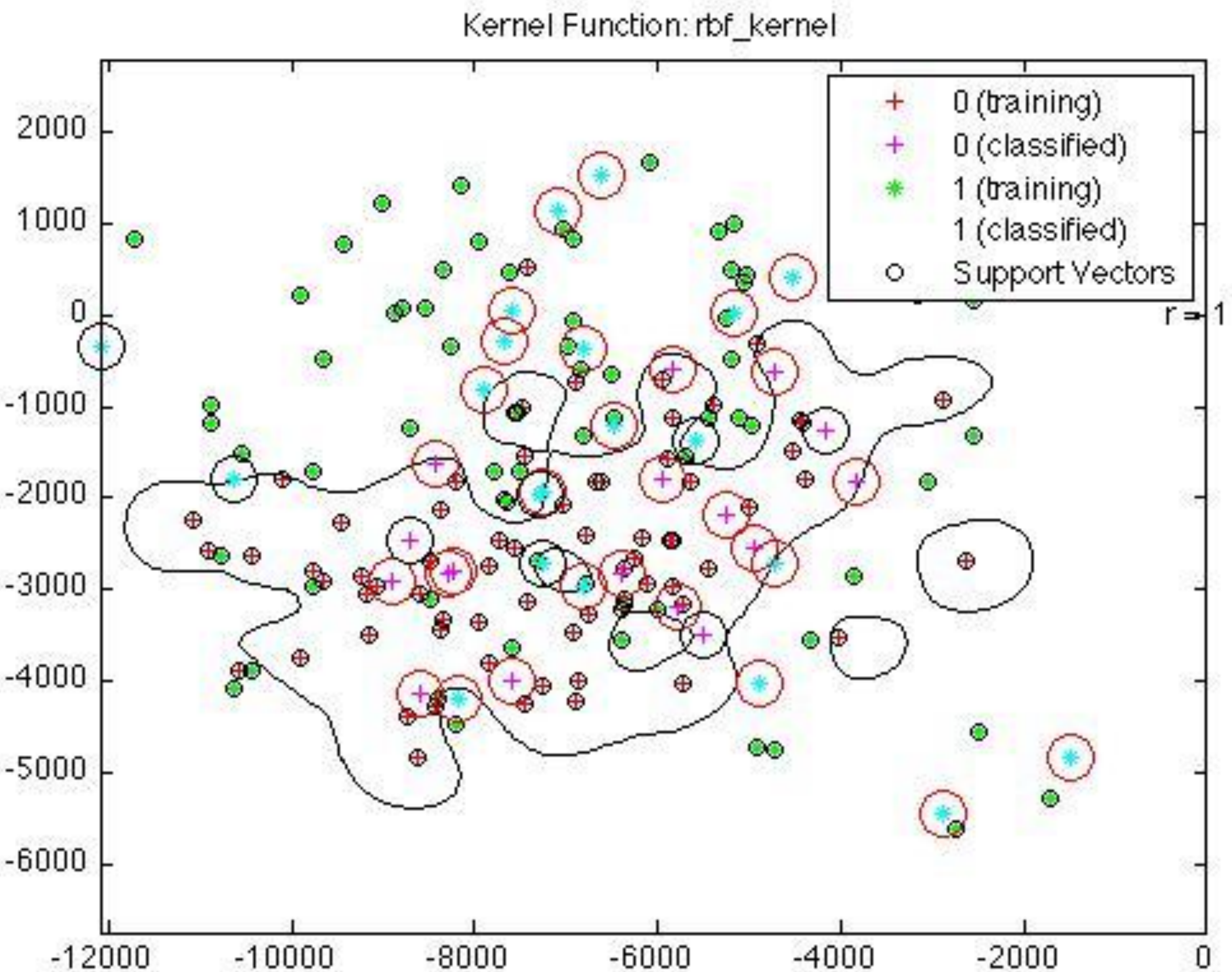
Four out of twenty misclassification using PCA SMV RBF

For 38 Testing Set, Three Cats classified as dogs, Five Dogs as Cats.



Kernel Function: rbf_kernel
-1 (training)
-1 (classified)
1 (training)
1 (classified)
Support Vectors
r = 1
3000
2000
1000
0
-1000
-2000
-3000
-4000
-5000
-6000
-12000
-10000
-8000
-6000
-4000
-2000
0

Thirty Eight Testing Set - Three Cats classified as dogs, Five Dogs as Cats

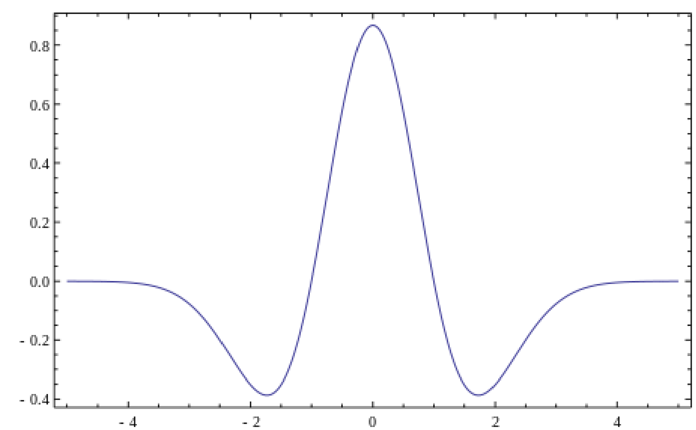


Edge Detection with Wavelet

Wavelet

- A **wavelet** is an oscillation with an amplitude that starts out at zero, increases and then decreases back to zero.
- Many types of transforms:
  - Continuous and Discrete
  - Father and Mother (also have children)

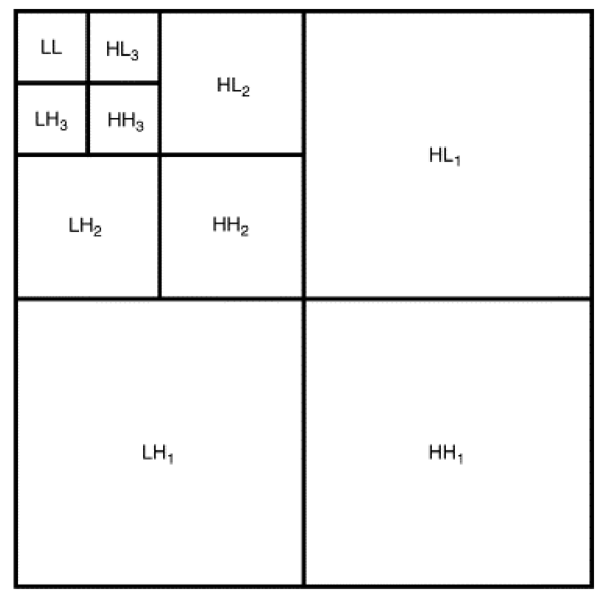
Example of a continuous wavelet:
Hat Wavelet



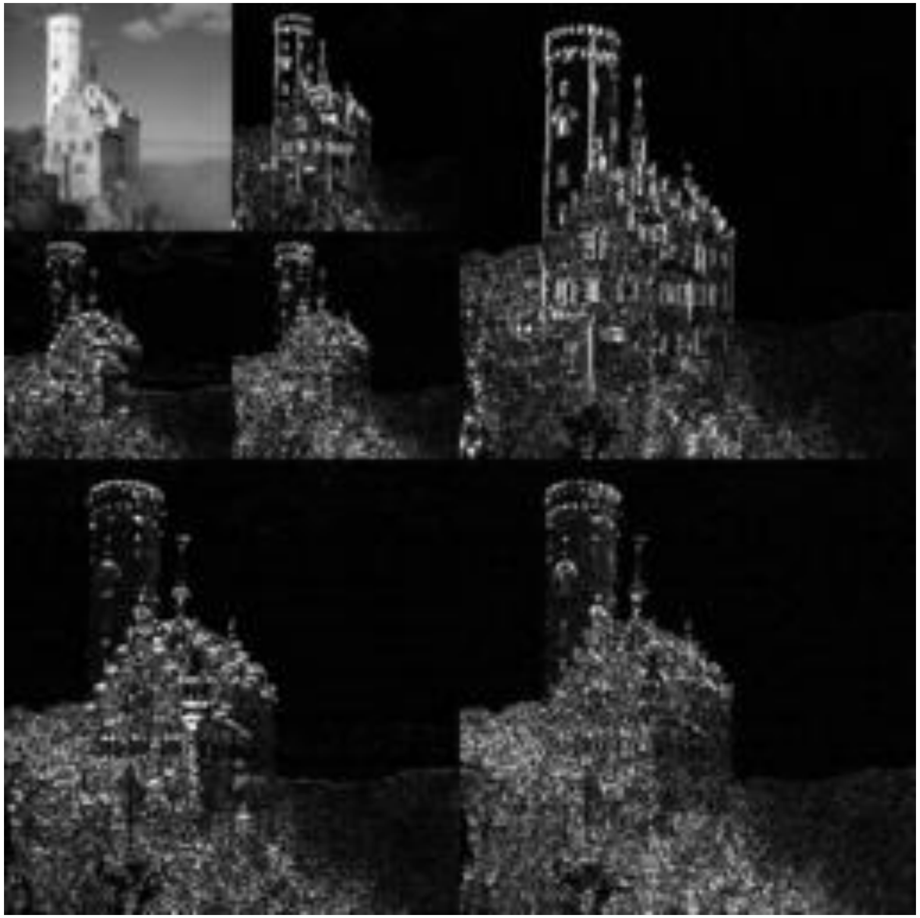
Continuous or Discrete

- When we perform a wavelet transform on an image, we use the 2D Discrete Wavelet.

L = *low-pass*
H = *high-pass*

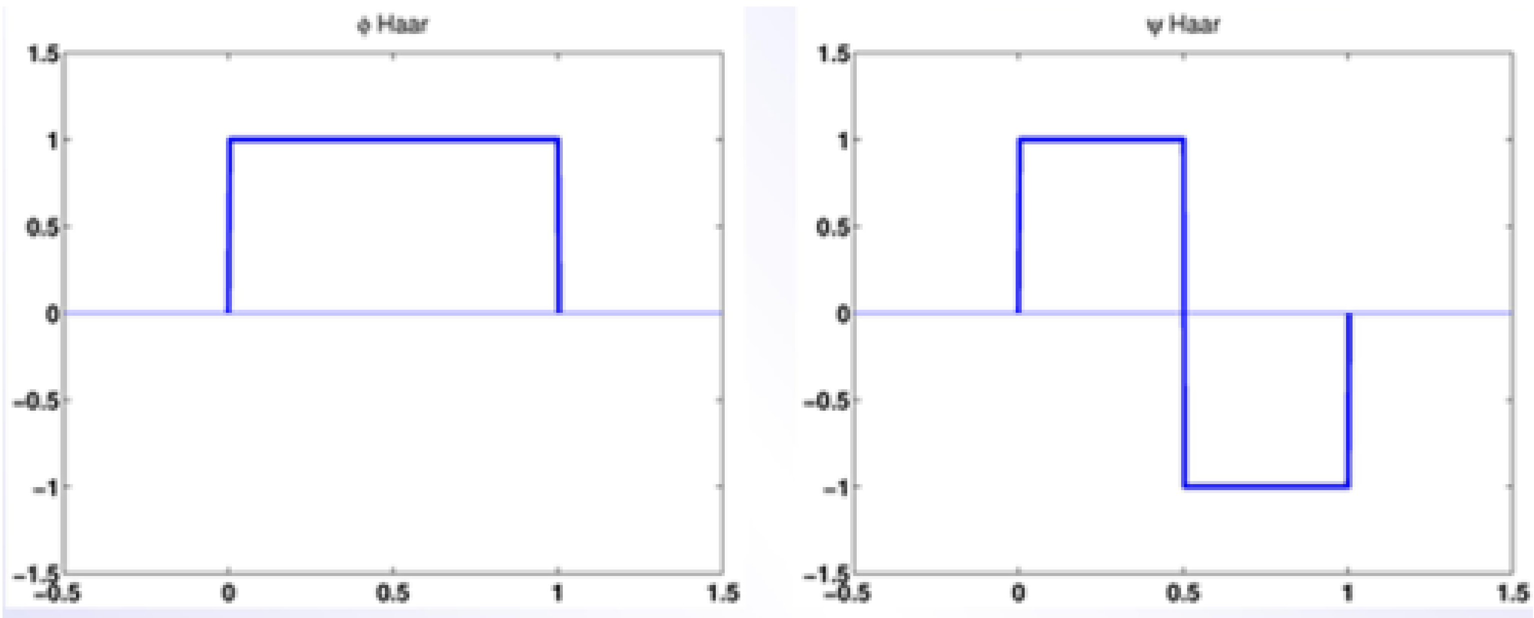


Pyramidal Decomposition

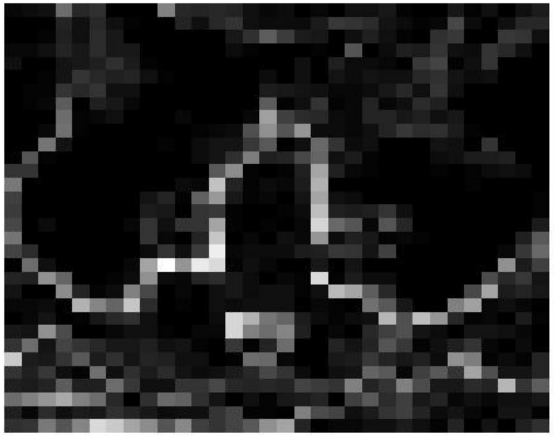


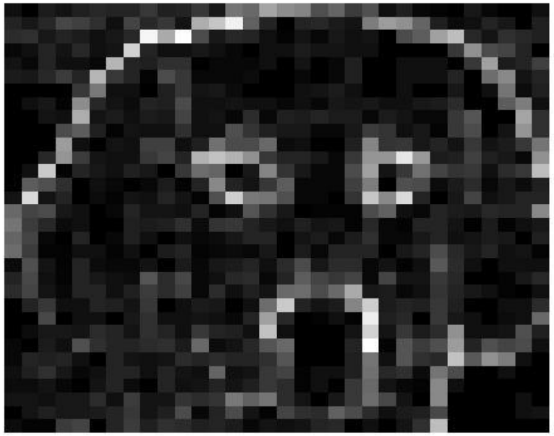
Haar Wavelet

- What is the Haar Wavelet?
  - A sequence of rescaled “square-shaped” functions which together form a basis



- Returning to Cat and Dog images:
  - Used one iteration of the Haar Wavelet for edge detection on Cat and Dog images
  - In order to recover the edges we use:
    - Edges = HL + LH
  - Results:





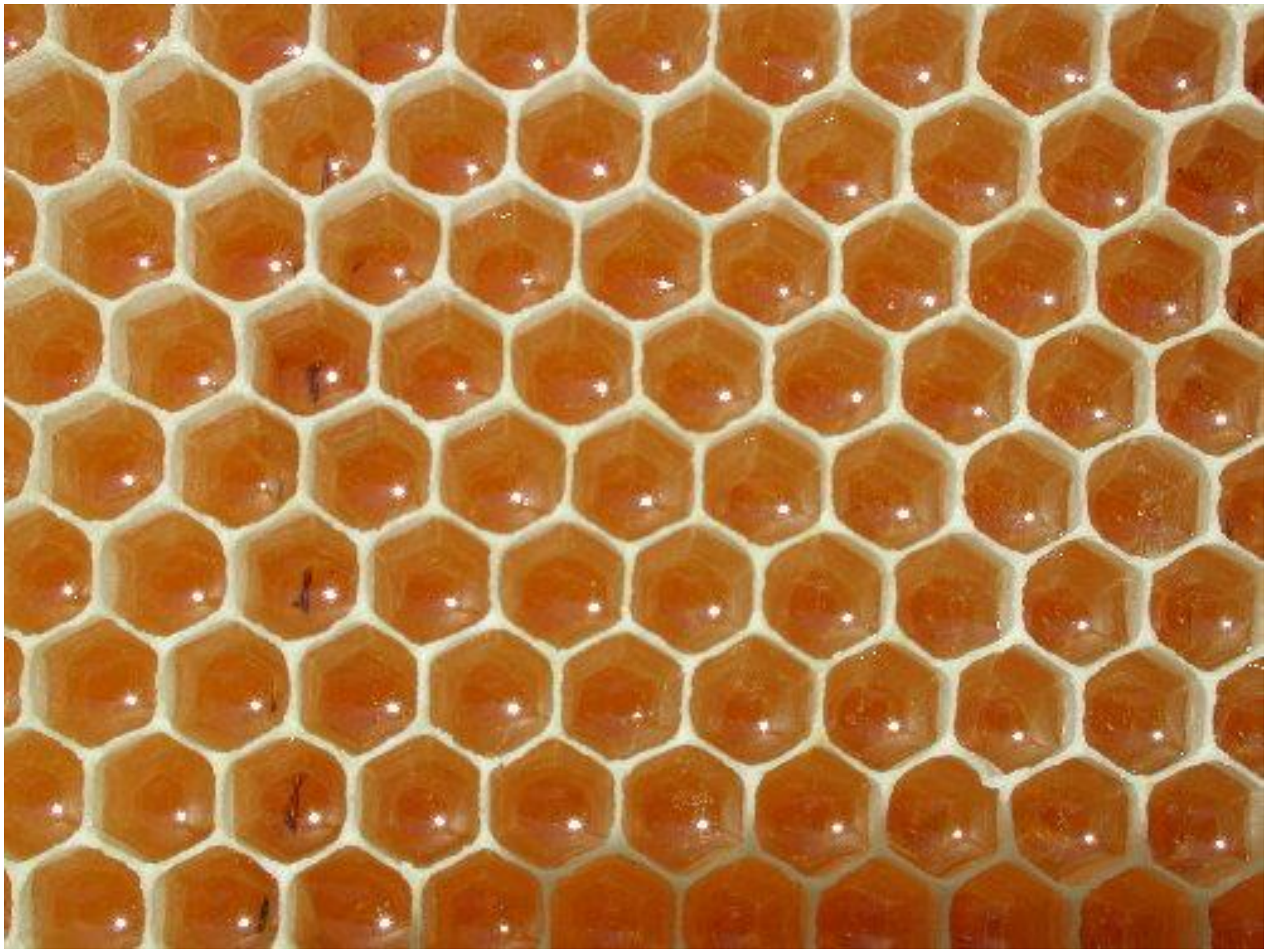
Classification with Voronoi Tessellation

Why so serious?
Georgy Voronoy

What is a tessellation?

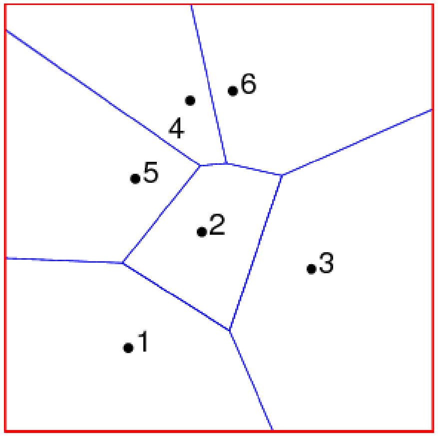
- **Tessellation** is the process of creating a two-dimensional plane using the repetition of a geometric shape with no overlaps and no gaps.
  - NOTE: Generalizations to higher dimensions are also possible.
- What is a tessellation that occurs in nature?

Correct Answer: Honeycomb

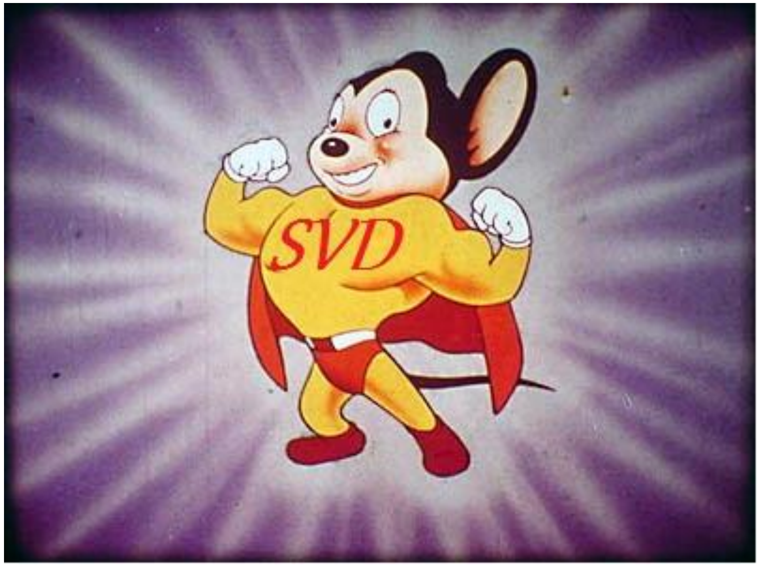


- Why a tessellation?

- Well, Voronoi Tessellation in 2-dimensions is a partitioning of a plane with *n* points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other.
  - This idea can be expanded to *n*-dimensions



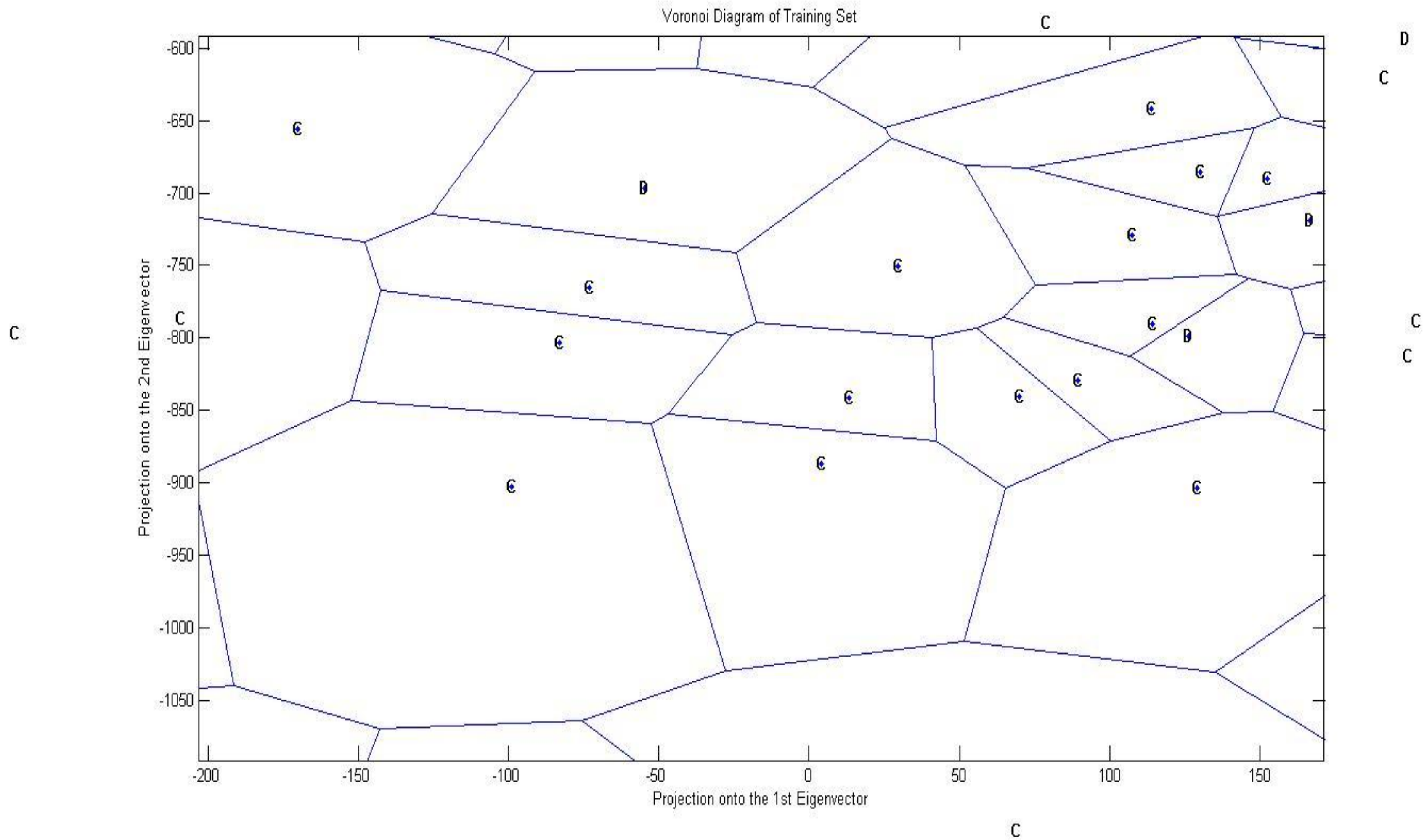
- We concatenate each image into column vectors.
- How can we represent each image as a point in 2-dimensions?

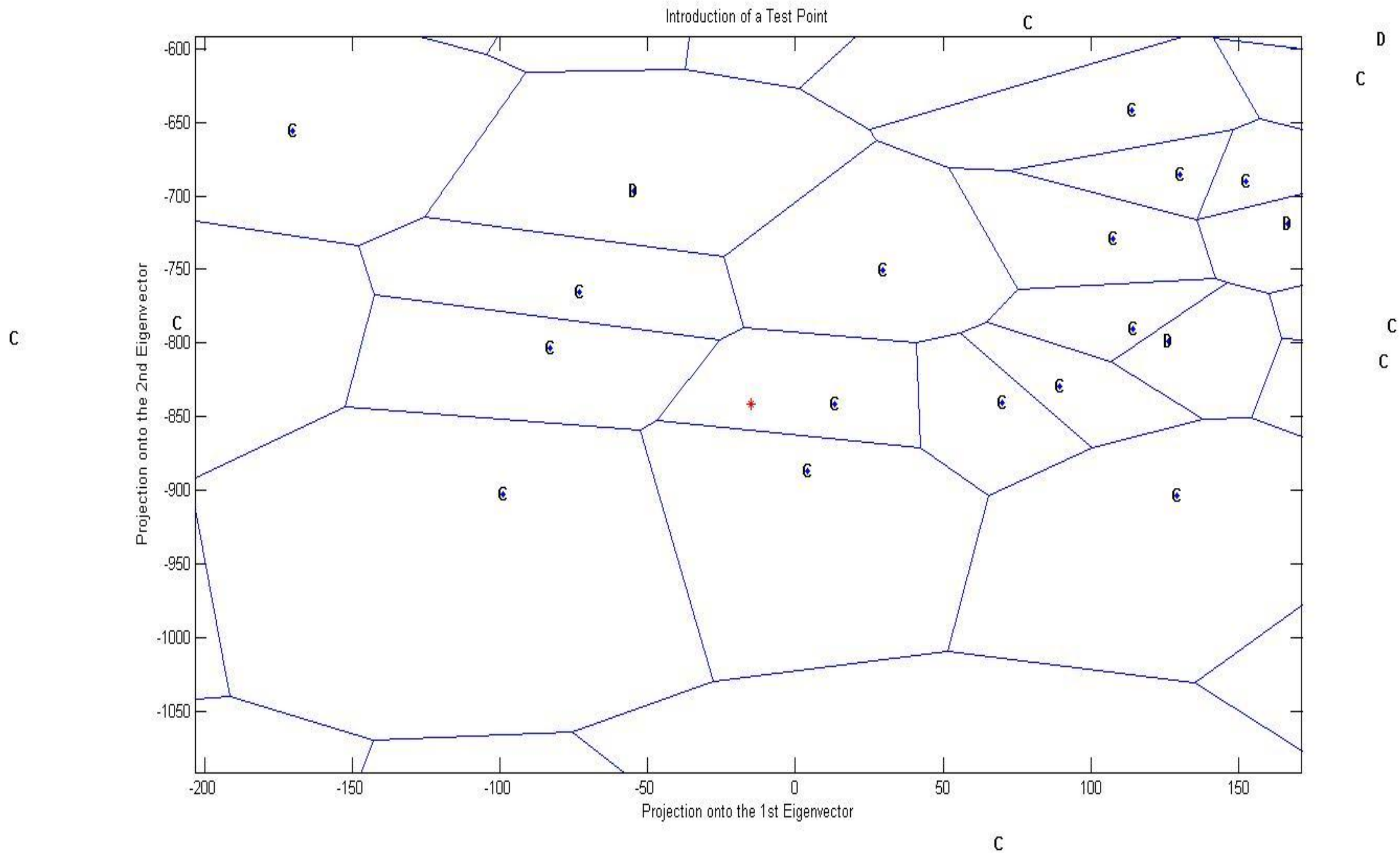


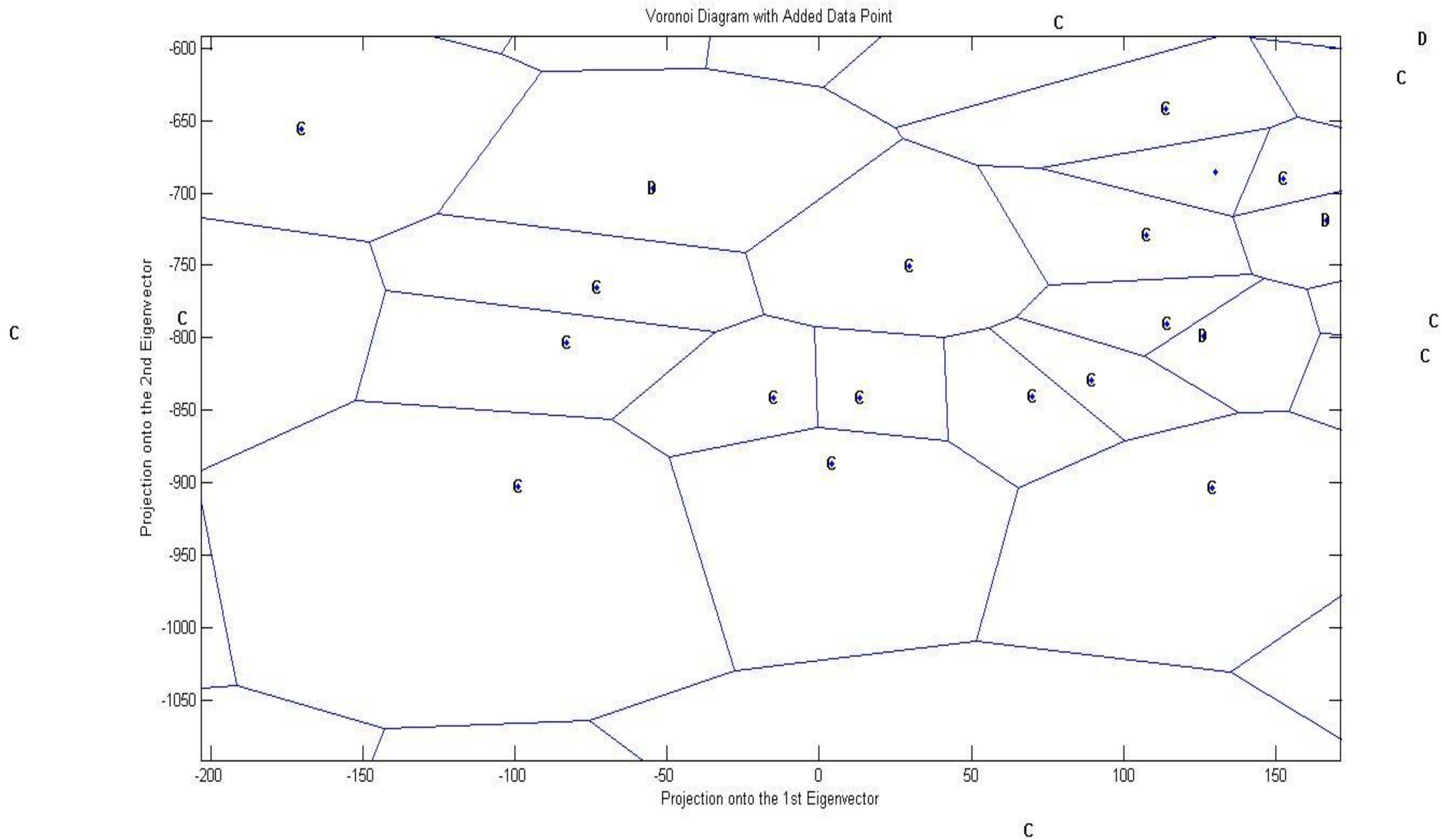
- After we take the SVD of the training set matrix, we use the two eigenvectors corresponding to the two highest eigenvalues.

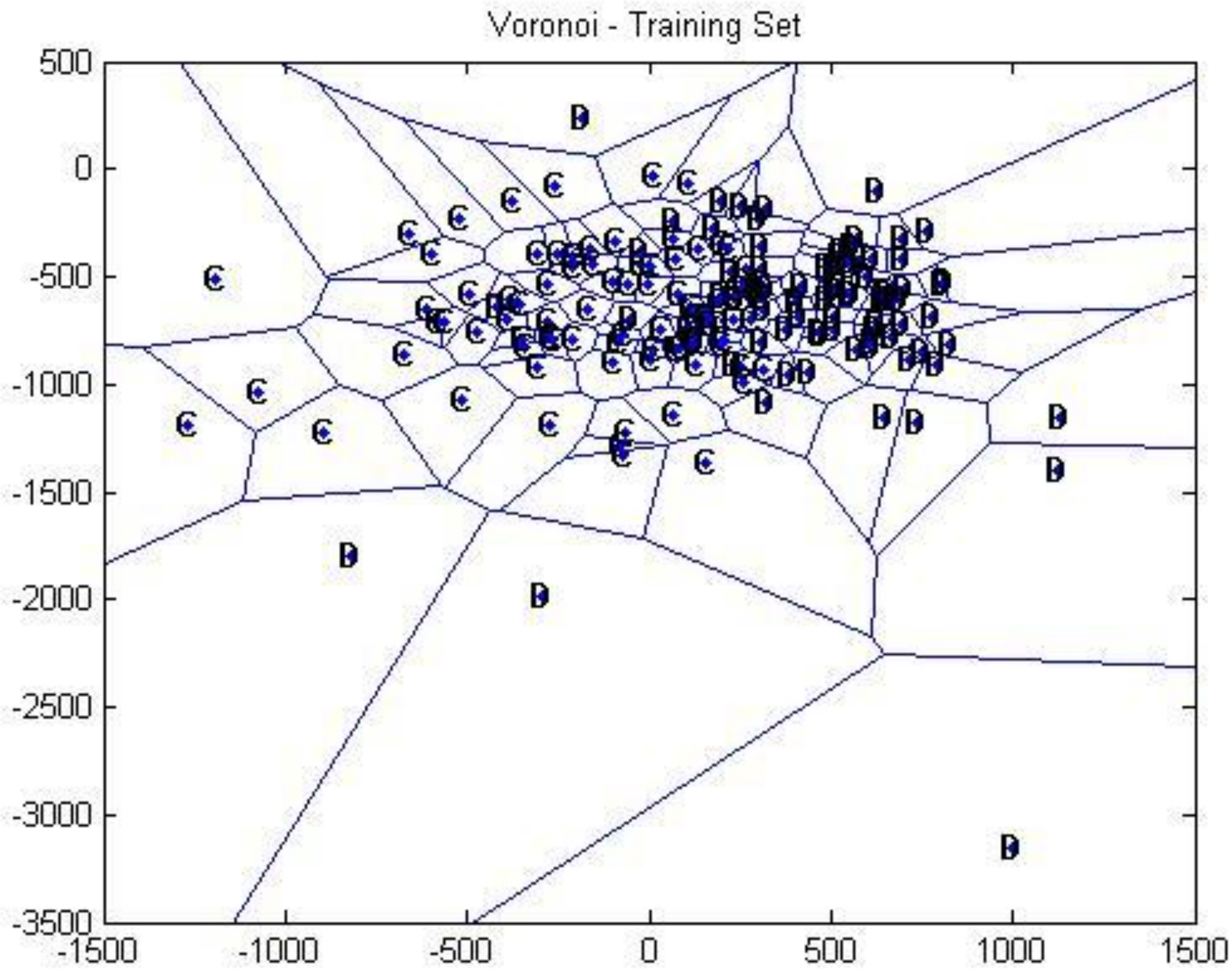
  - Retains majority of data

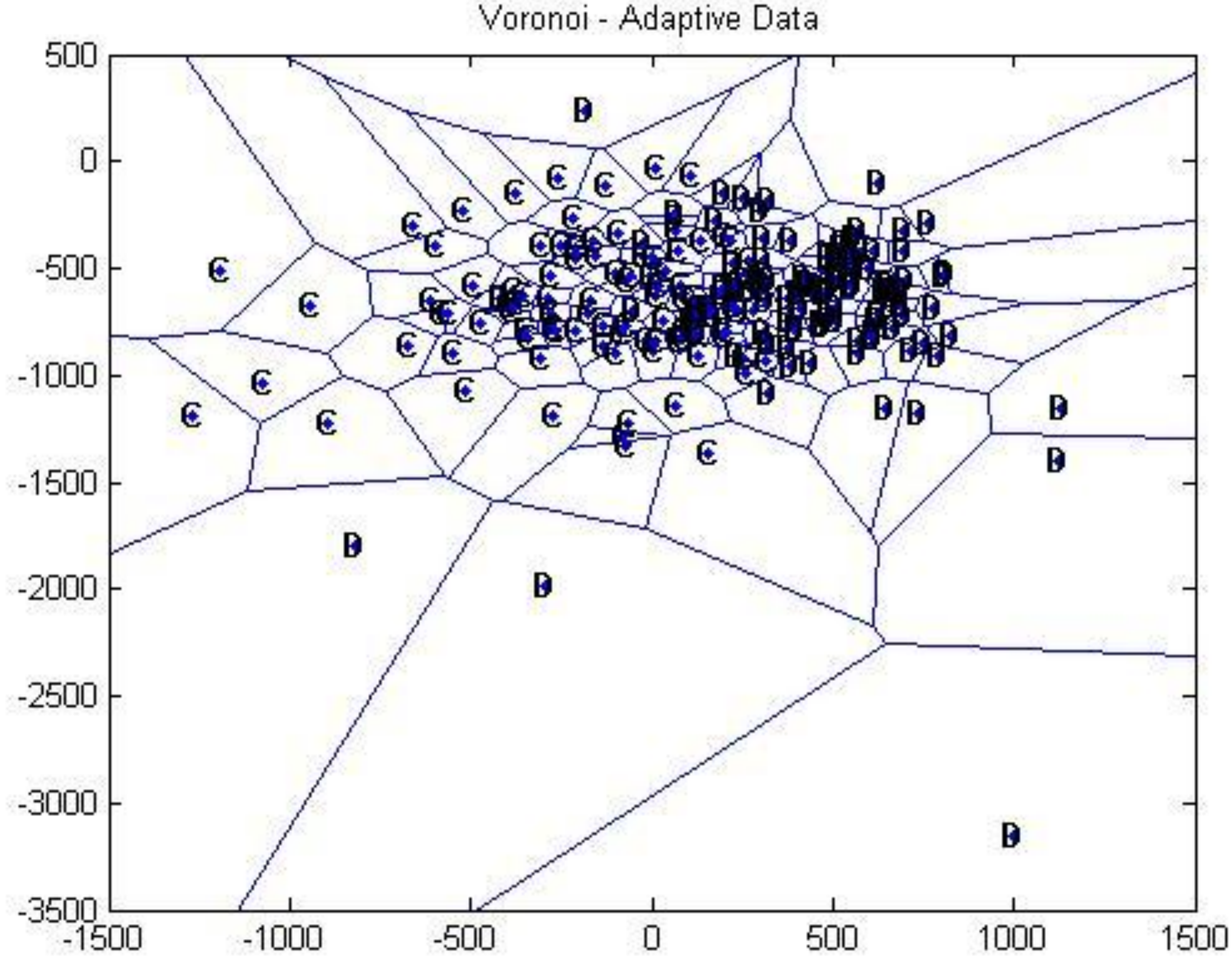
  - Project each concatenated image onto the two eigenvectors


Voronoi Diagram of Training Set
Projection onto the 2nd Eigenvector
Projection onto the 1st Eigenvector


Introduction of a Test Point
Projection onto the 2nd Eigenvector
-600
-650
-700
-750
-800
-850
-900
-950
-1000
-1050
-200
-150
-100
-50
0
50
100
150
Projection onto the 1st Eigenvector


Voronoi Diagram with Added Data Point
Projection onto the 2nd Eigenvector
-600
-650
-700
-750
-800
-850
-900
-950
-1000
-1050
-200
-150
-100
-50
0
50
100
150
Projection onto the 1st Eigenvector


Voronoi - Training Set
500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
-1500
-1000
-500
0
500
1000
1500



Voronoi - Adaptive Data
500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
-1500
-1000
-500
0
500
1000
1500

Classification Rate

- Cat Classification: 17 out of 19
  - 89.5 %

- Dog Classification: 17 out of 19
  - 89.5%